Assignment - 4
(Object-Oriented Structure Measurement)

Course Code	:	SE 3204
Course Title	:	Software Metrics

Submitted To:

Dipok Chandra Das
Assistant Professor, IIT
Noakhali Science and Technology University (NSTU)

Team Members:

1.	Md. Armanur Rashid	ASH1925013M
2.	Sourav Debnath	ASH1925022M
3.	Sourav Barman	ASH1925030M

Noakhali Science and Technology University (NSTU)

Table of Contents

Table of Figures

1. Measuring Coupling in Object-Oriented System

Coupling is the degree of a measure of how closely connected two routines or modules are; the strength of the relationships between modules.

We can measure coupling in object-oriented system by

1. Coupling Between Object Class (CBO)
2. Response For Class (RFC)
3. Message Passing Coupling (MPC)

1.1 Coupling between Object Class

Coupling between objects (CBO) is a count of the number of classes that are coupled to a particular class i.e. where the methods of one class call the methods or access the variables of the other.

Measurement Process: We consider here

1. Methods Call
2. Class Extends

Table 1: Coupling between Object

Class	Number	Coupled WIth
Aboutus	2	Panel_BackButton_Template Frame_Container_Template
Back_Button	1	Home
Basic_Frame_Duplicity	1	Frame_Container_Template
ButtonSound	0	
Capitalization	6	Panel__BackButton_Template Frame_Container_Template Back_Button ButtonSound save_file file_read
Check_extension	1	ButtonSound
Duplicity_Folder	5	Basic_Frame_Duplicity ButtonSound Back_Button save_file Sorting_result
Duplicity_Random_File	5	Basic_Frame_Duplicity ButtonSound Back_Button

		save_file Sorting_result
File_read	2	ButtonSound Check_extension
Frame_Container_Template	0	
Highlight_text	0	
Home	7	sound_button Duplicity_Random_File Duplicity_Folder Capitalization Spell_Check Aboutus User_GuideLines
MainClass	1	Home
Panel__BackButton_Template	1	Frame_Container_Template
Save_file	0	
Sorting_result	0	
Spell_Check	4	Panel__BackButton_Template Frame_Container_Template Back_Button file_read
SplashScreen	1	Home
User_GuideLines	1	ButtonSound

1.2 Response for Class

This is the size of the Response set of a class. The Response set for a class is defined by a set of methods that can potentially be executed in response to a message received by an object of that class. That means all the methods in the class and all the methods that are called by methods in that class.

Measurement Process: we count here number of methods in the class and number of distinct method calls made by the methods of this call.

RFC = number of methods in the class + number of distinct method calls made by the methods.

Aboutus: 6

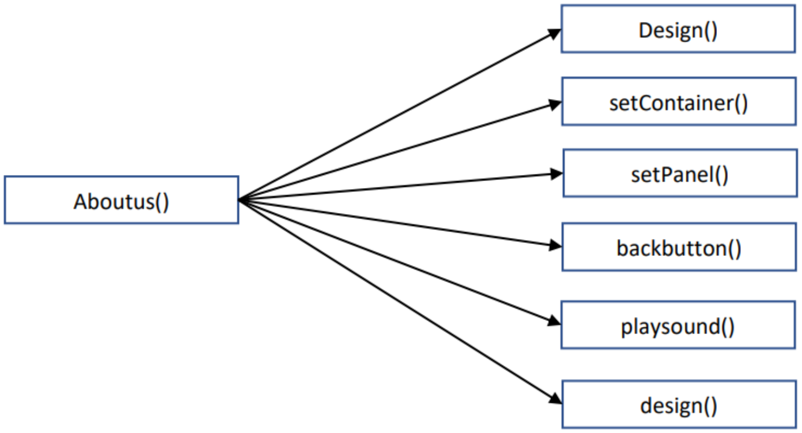

Figure 1: RFC (Aboutus)

Back_Button: 1

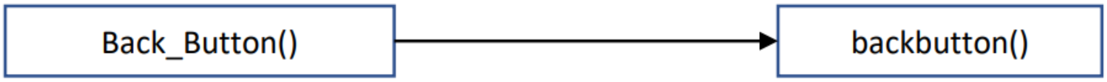

Figure 2: RFC (Back_Button)

Basic_Frame_Duplicity: 5

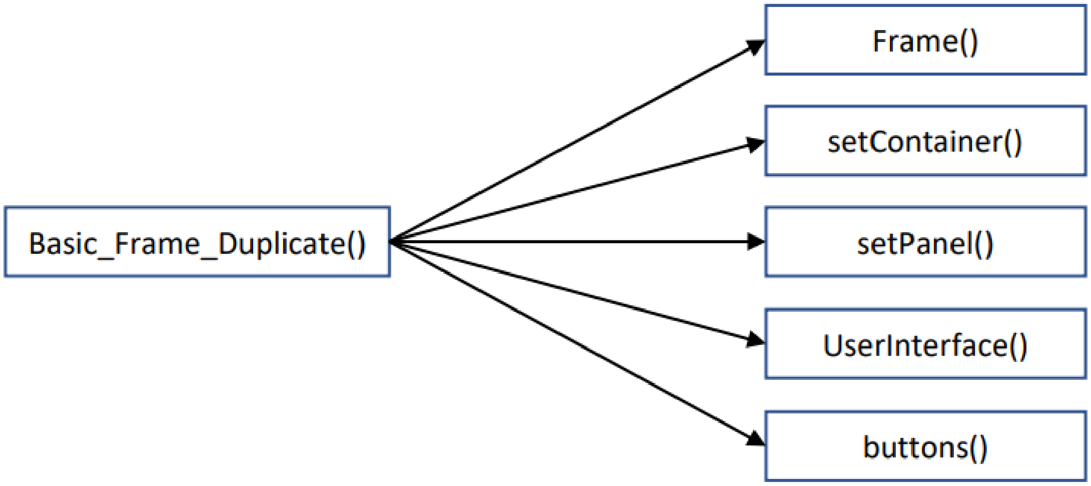

Figure 3: CBO (Basic_Frame_Duplicity)

ButtonSound: 1

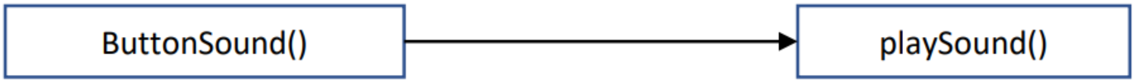

Figure 4: RFC (ButtonSound)

Capitalization: 13

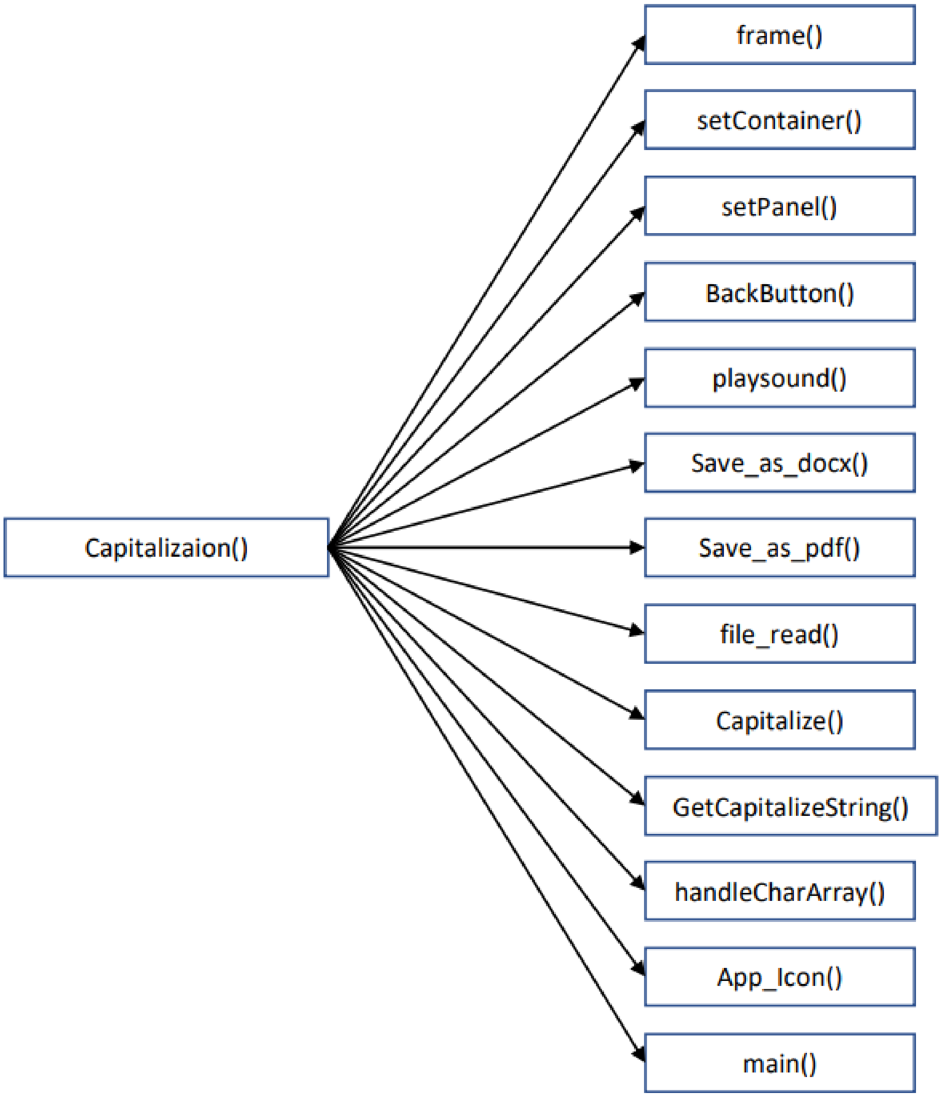

Figure 5: RFC (Capitalization)

Check_Extension: 2

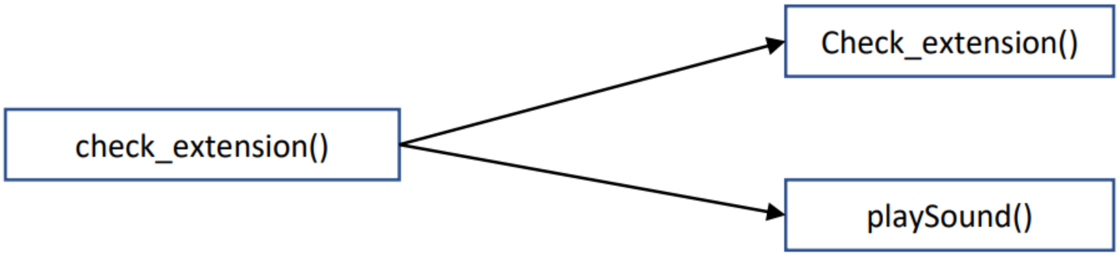

Figure 6: RFC (Check_Extension)

Duplicity_Folder: 19

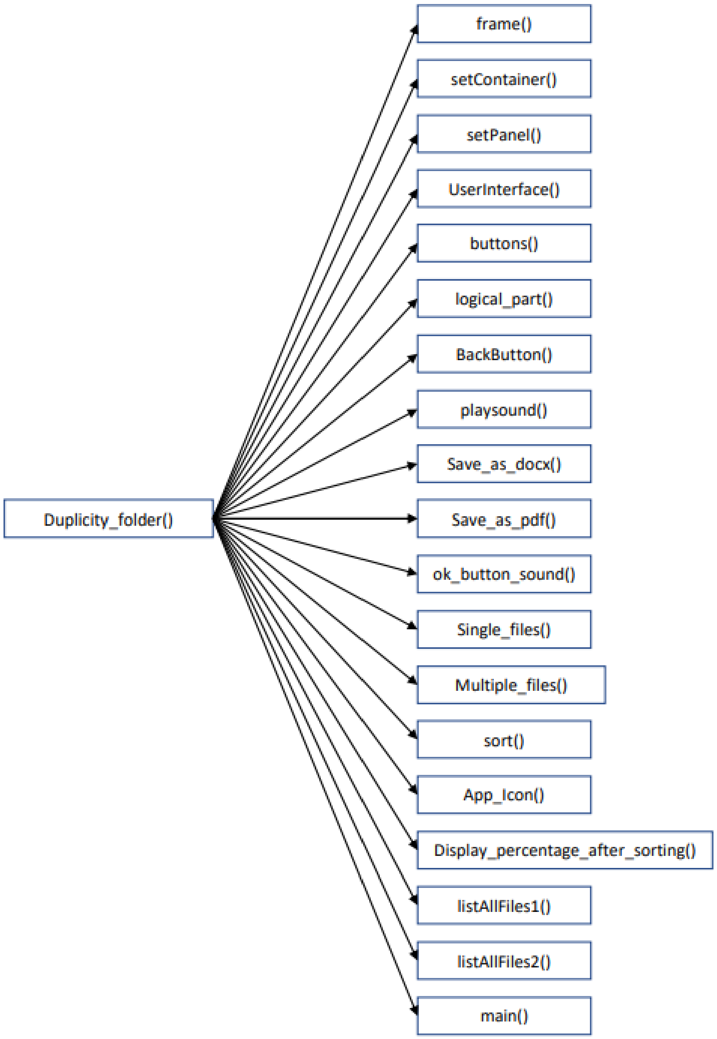

Figure 7: RFC (Duplicity_Folder)

Duplicity_Random_File: 16

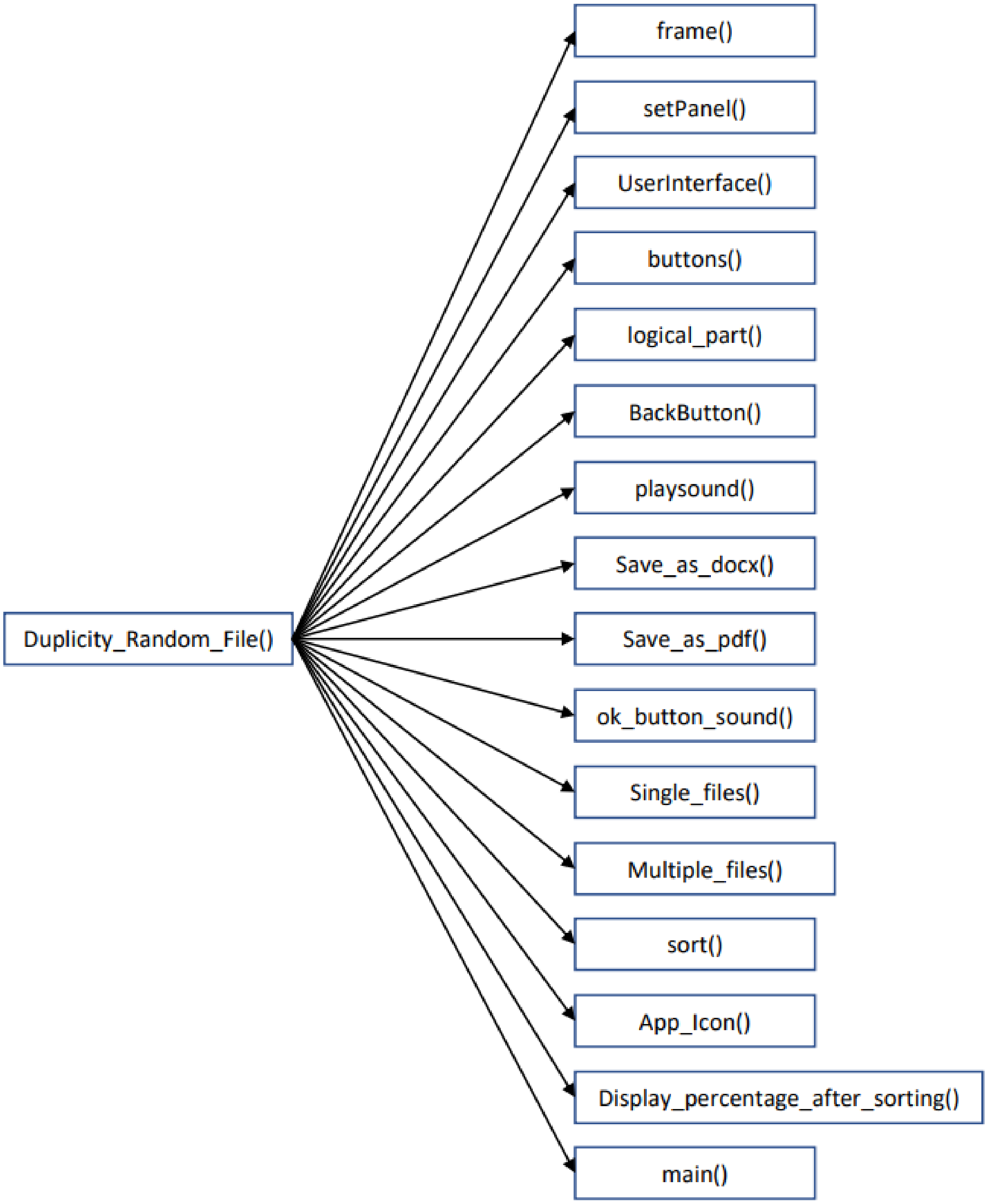

Figure 8: RFC (Duplicity_Random_File)

fileRead: 1

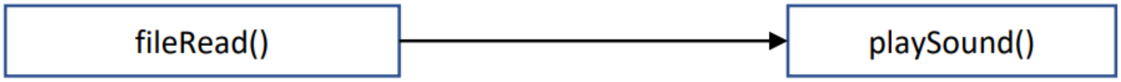

Figure 9: RFC (fileRead)

Frame_Container_Template: 2

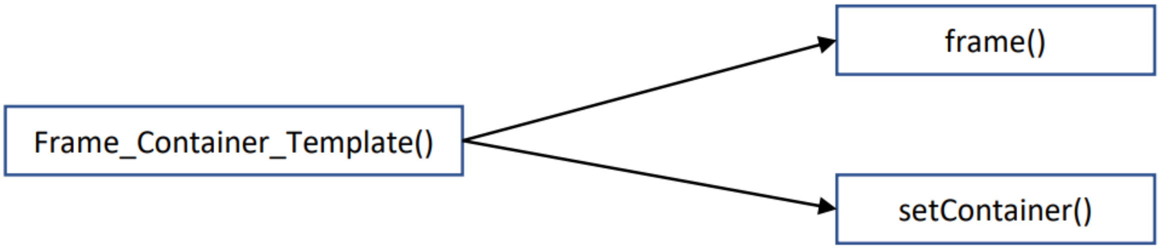

Figure 10: RFC (Frame_Container_Template)

highlight: 1

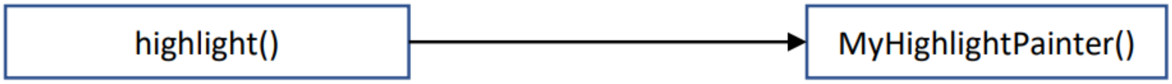

Figure 11: RFC (highlight)

Home: 4

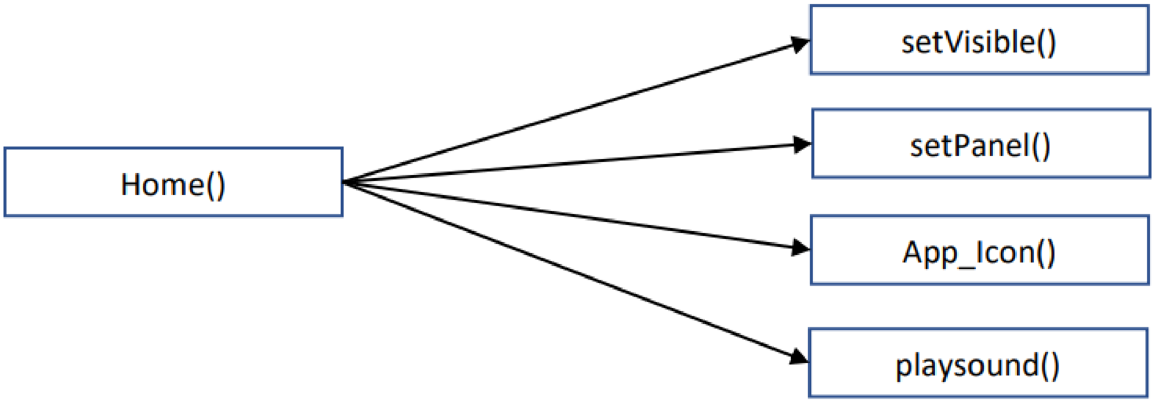

Figure 12: RFC (Home)

MainClass: 1

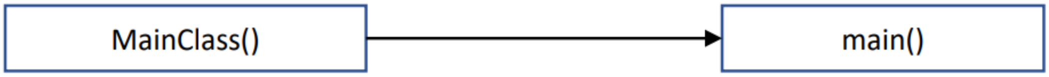

Figure 13: RFC (MainClass)

Panel_BackButton: 4

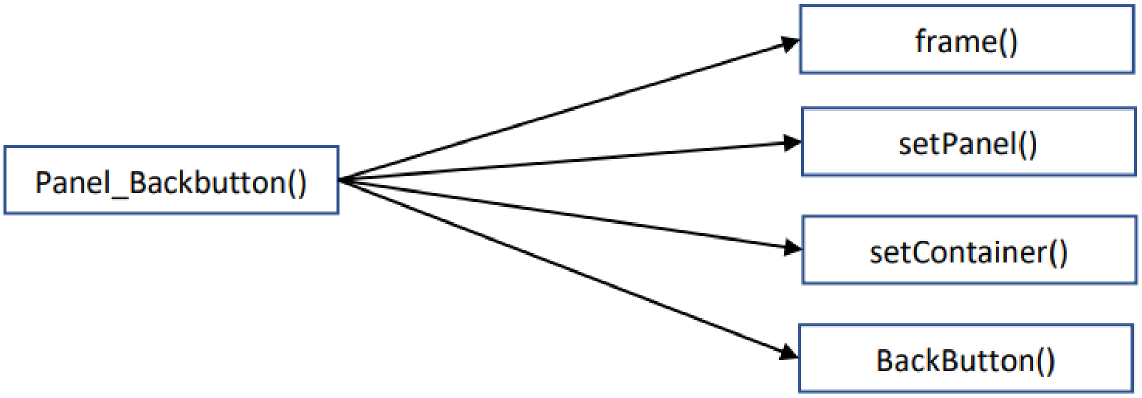

Figure 14: RFC (Panel_BackButton)

Save_file: 2

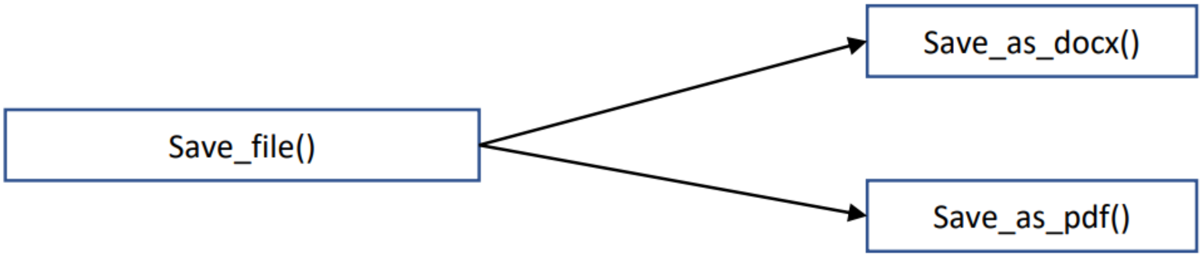

Figure 15: RFC (Save_file

Sorting_result: 1

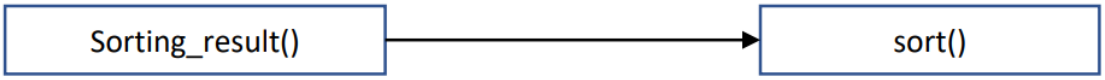

Figure 16: RFC (Sorting_result)

Spell_Check: 11

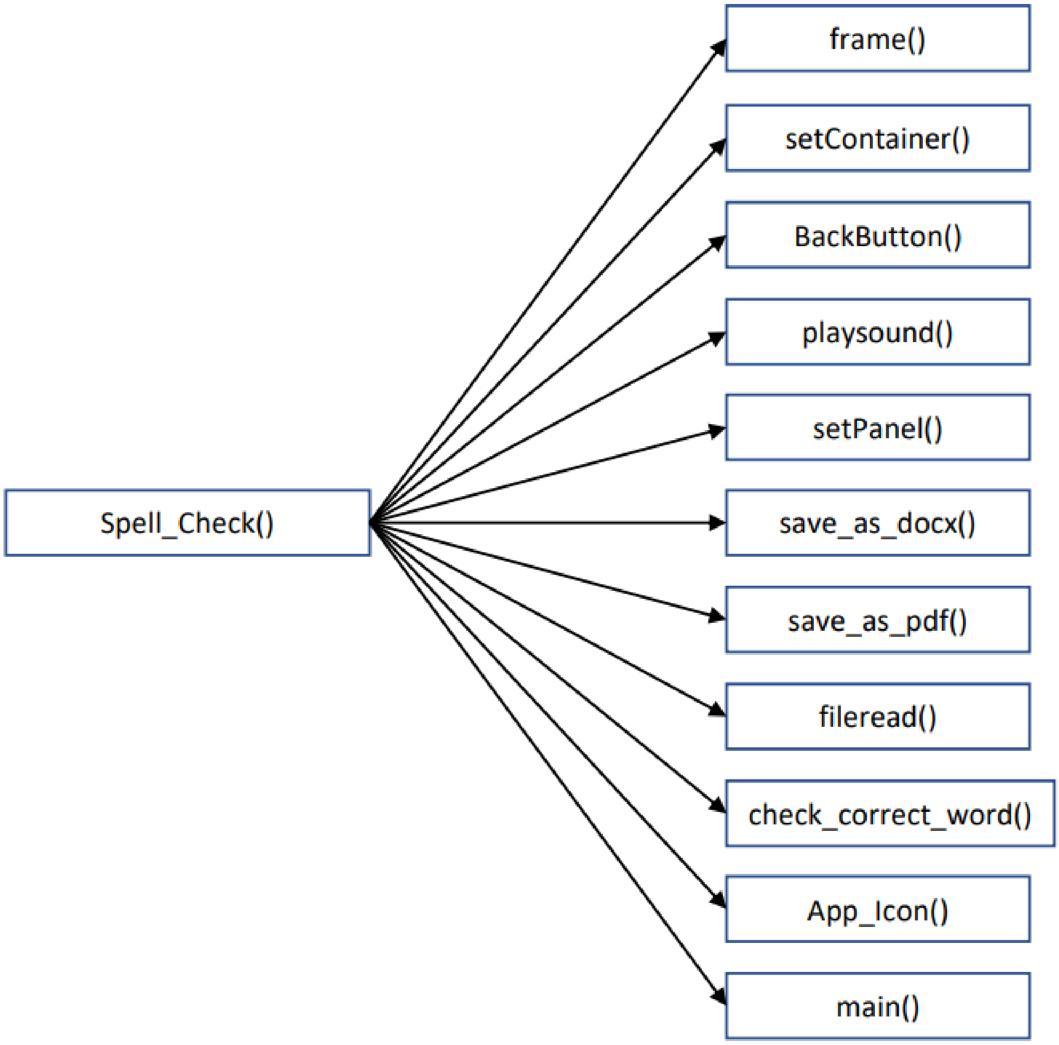

Figure 17: RFC (Spell_Check)

SplashScreen: 8

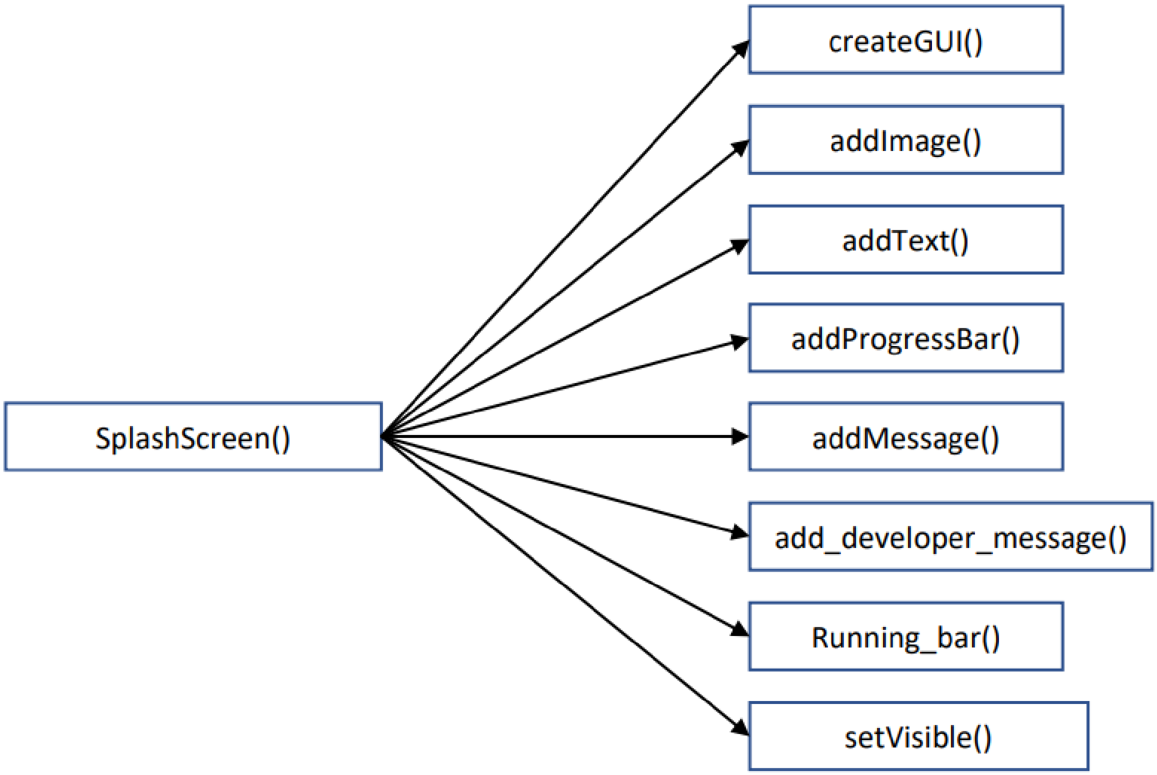

Figure 18: RFC (SplashScreen)

User_GuideLines: 6

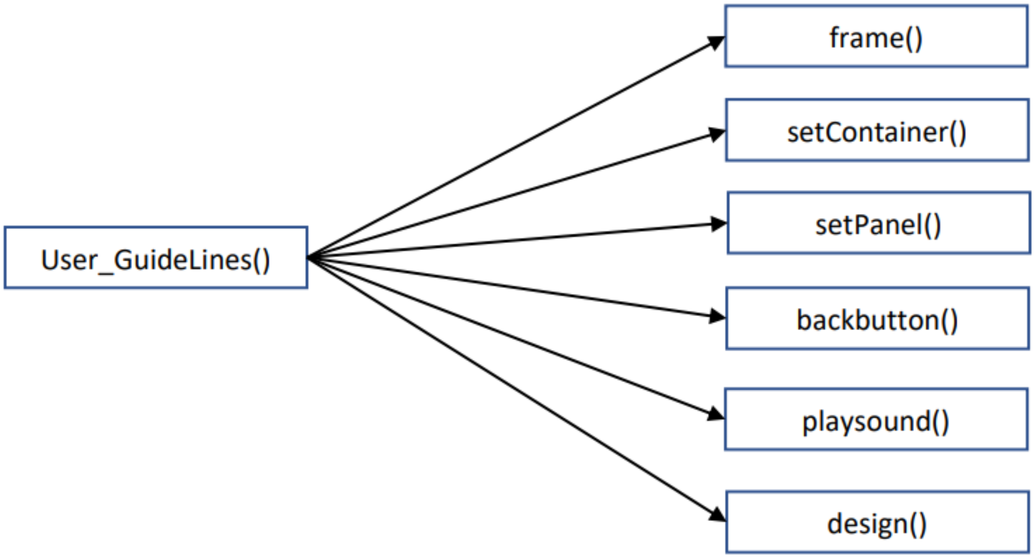

Figure 19: RFC (user_GuideLines)

1.3 Message Passing Coupling

The MPC measures the number of method calls defined in methods of a class to methods in other classes, and therefore the dependency of local methods to methods implemented by other classes.

Measurement Process: Number of distinct method calls made by the methods of this call.

Aboutus: 5

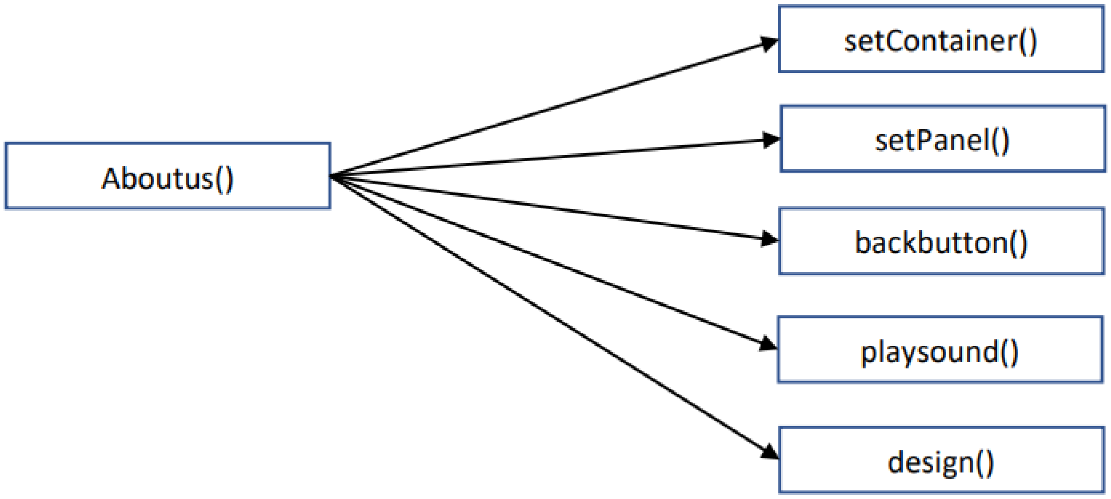

Figure 20: MPC (Aboutus)

Basic_Frame_Duplicity: 2

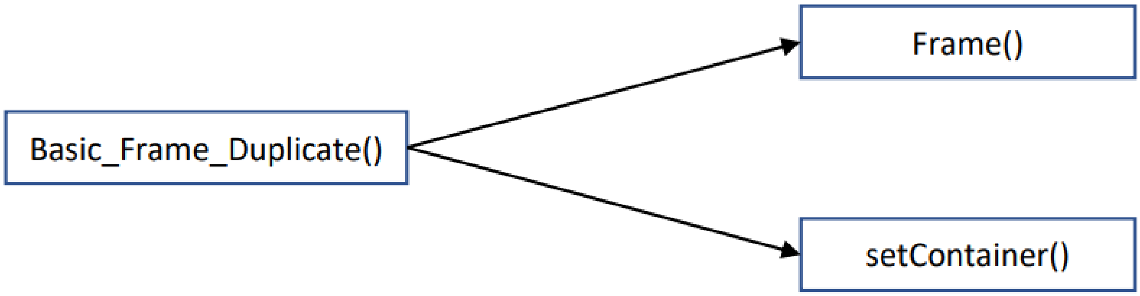

Figure 21: MPC (Basic_Frame_Duplicity)

Capitalization: 9

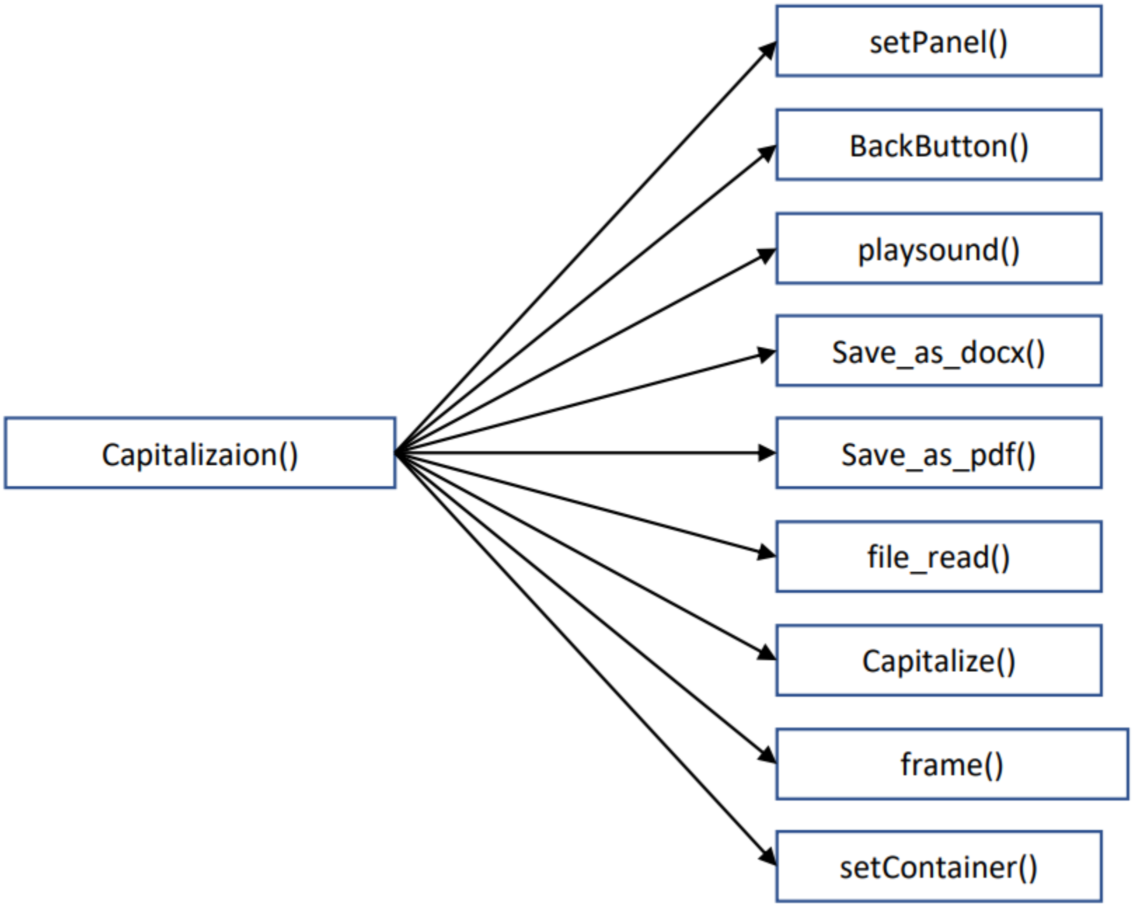

Figure 22: MPC (Capitalization)

Check_Extension: 1

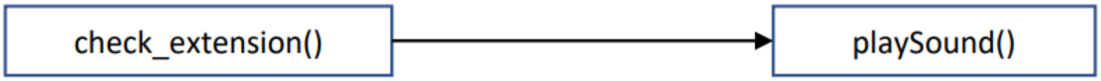

Figure 23: MPC (Check_Extension)

Duplicity_Folder: 10

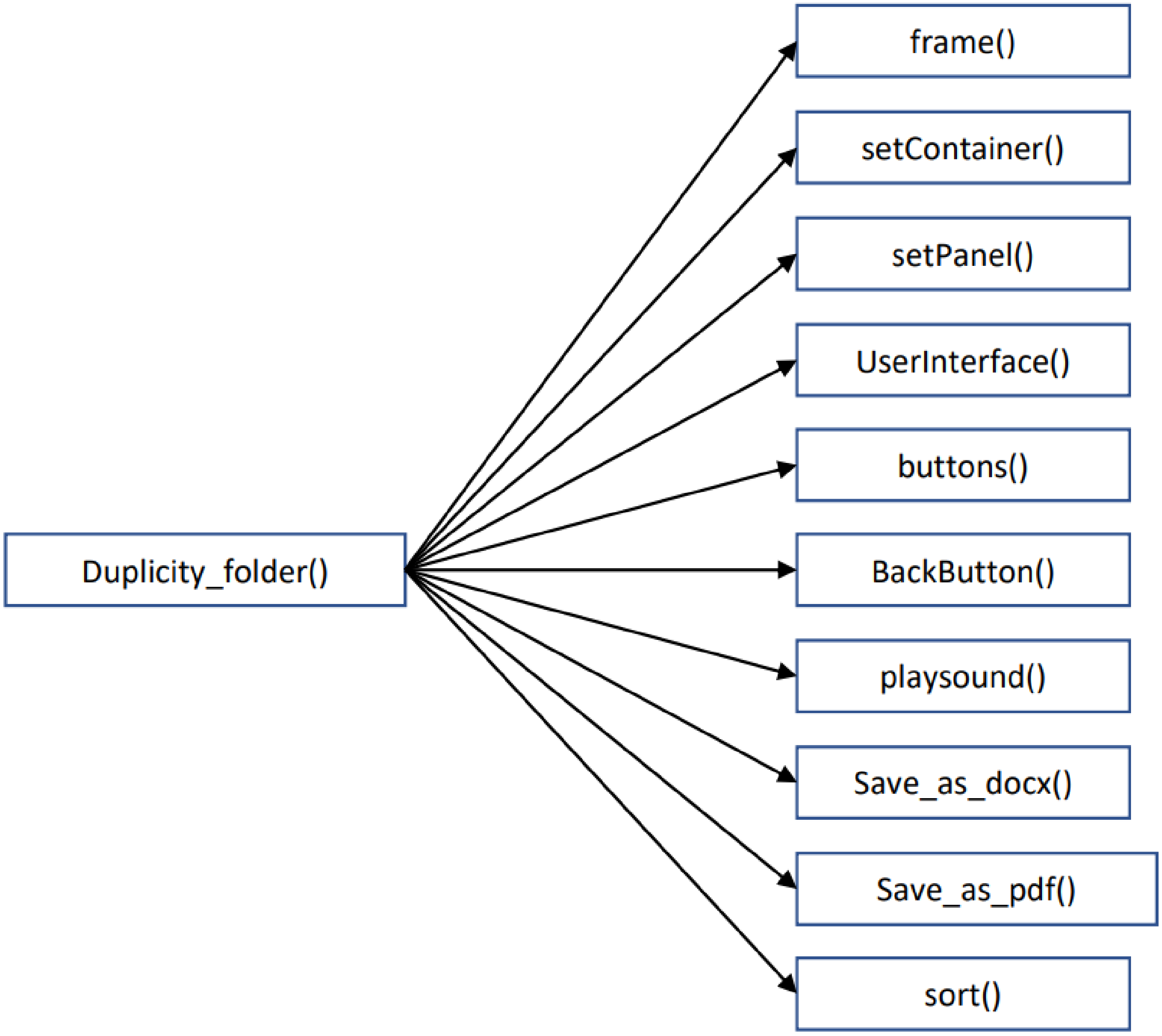

Figure 24: MPC (Duplicity_Folder)

Duplicity_Random_File: 9

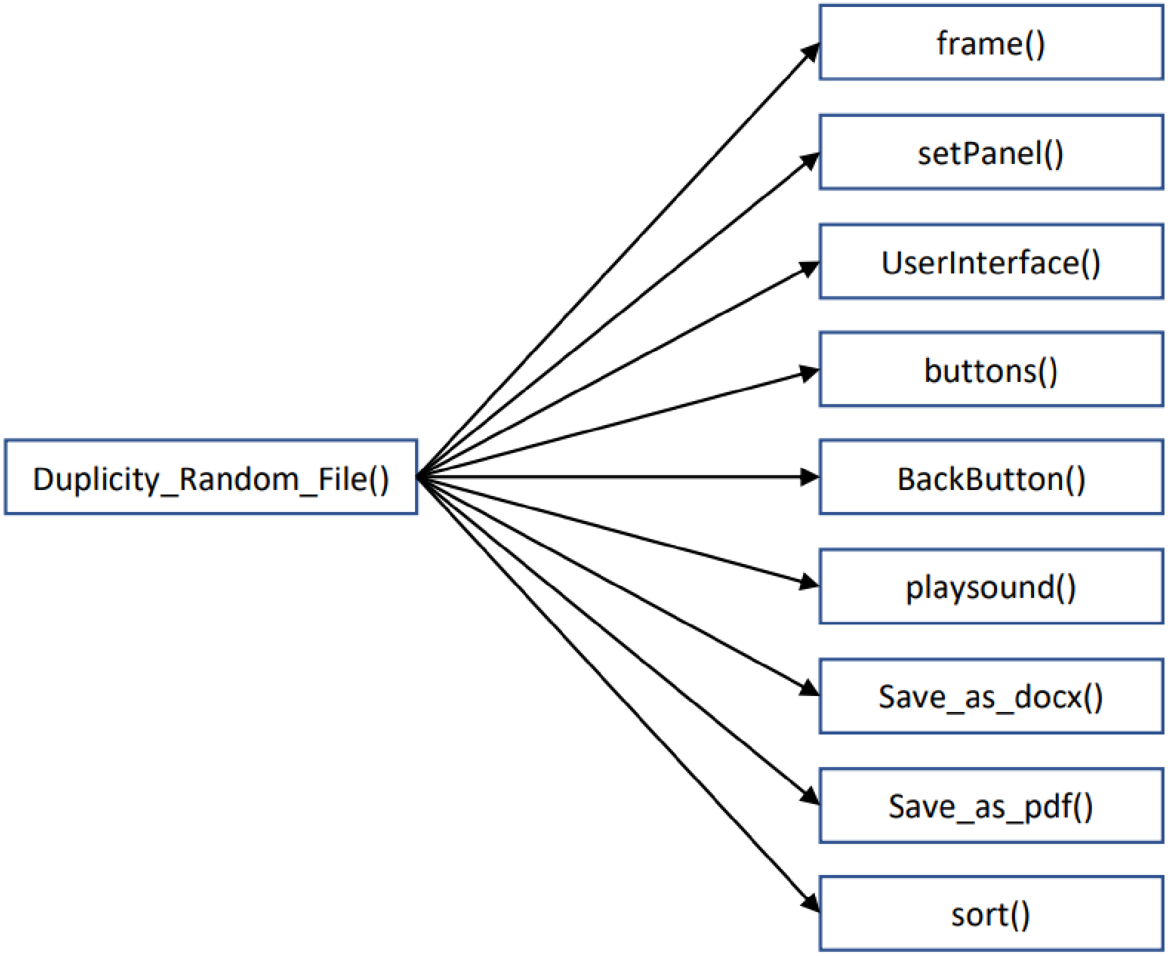

Figure 25: MPC (Duplicity_Random_File

fileRead: 1

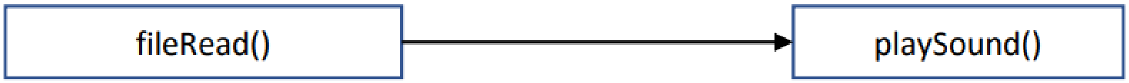

Figure 26: MPC (fileRead

Home: 2

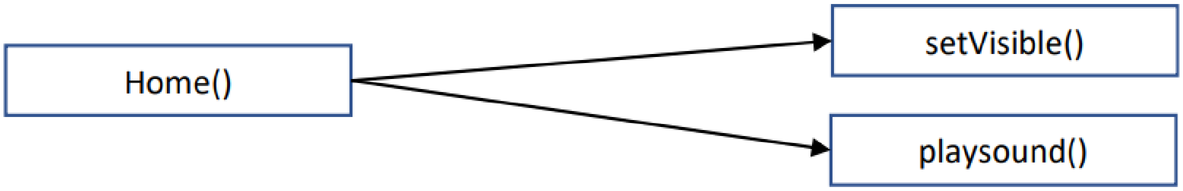

Figure 27: MPC (Home)

Panel_BackButton: 2

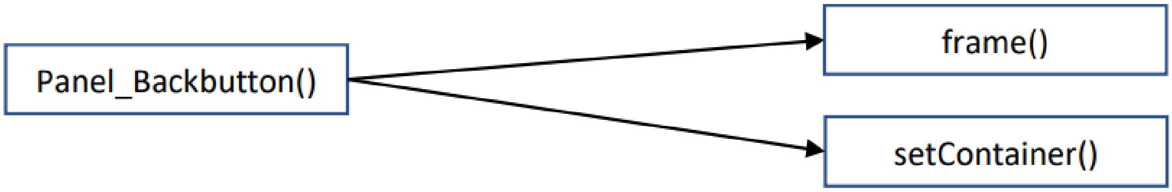

Figure 28: MPC (Panel_BackButton)

Spell_Check: 11

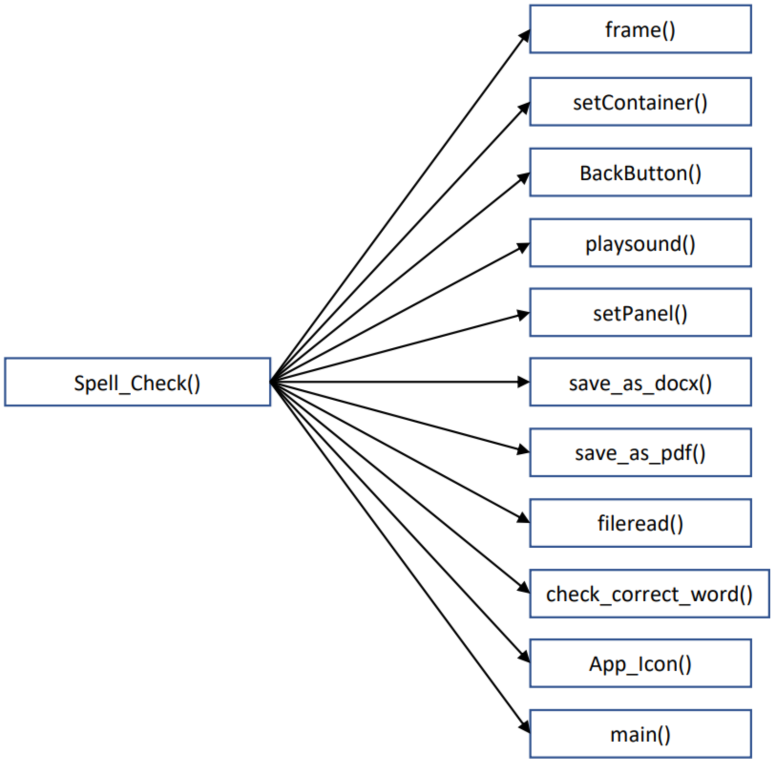

Figure 29: MPC (Spell_Check)

User_GuideLines: 5

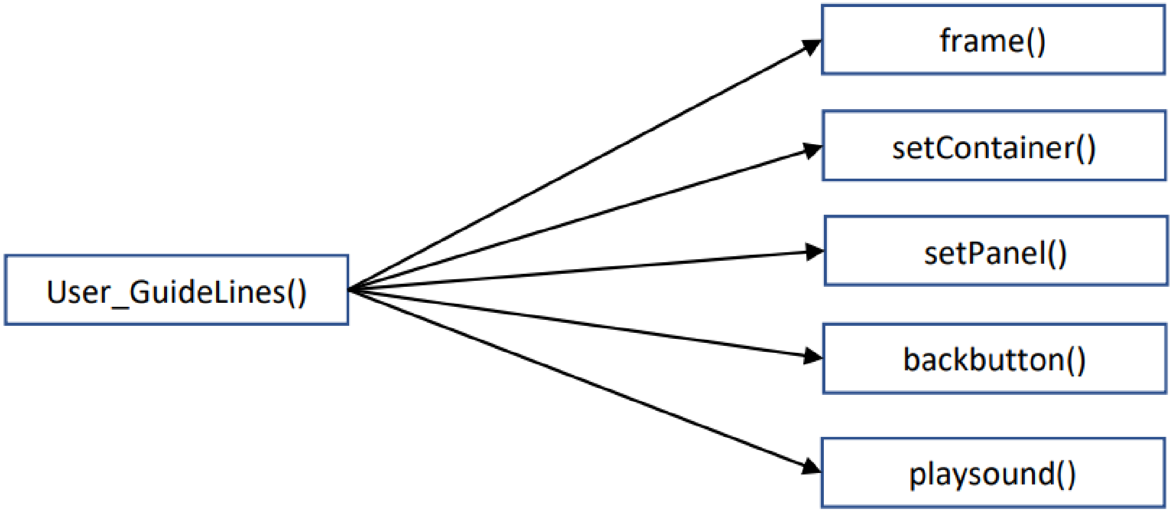

Figure 30: MPC (User_GuideLines)

2. Measuring Cohesion in Object-Oriented System

Cohesion metrics measure how well the methods of a class are related to each other. cohesion refers to the degree to which the elements inside a module belong together. We can measure Cohesion in Object-Oriented System by LCOM (Lack of Cohesion Metric)

Measurement Process:

LCOM = 1 – (sum(MF) / M*F)

Where,

- M is the number of methods in class
- F is the number of instance fields in the class.
- MF is the number of methods of the class accessing a particular instance field.
- Sum(MF) is the sum of MF over all instance fields of the class.

Table 2: Measuring Cohesion (LCOM)

Class	**LCOM**
Aboutus	0
Back_Button	-1
Basic_Frame_Duplicity	0.5
ButtonSound	0
Capitalization	0.2
Check_extension	-1
Duplicity_Folder	0.153
Duplicity_Random_File	0.153
File_read	-1
Frame_Container_Template	1
Highlight_text	0
Home	.22
MainClass	-1
Panel__BackButton_Template	0.67
Save_file	-1
Sorting_result	0
Spell_Check	0.20
SplashScreen	0
User_GuideLines	0

3. Object-Oriented Length Measure

Generally, length measures indicate the distance from one element to another. In object-oriented systems, distances depend on the perspective and the model representing an appropriate view of the system. We can measure it by DIT (Depth Inheritance Tree).

The DIT metric tells us how deep a class is in an inheritance hierarchy. The DIT metric is the length of the maximum path from the node to the root of the tree.

Measurement Process: It is done by looking at inheritance tree by looking which class extends or implements which class. We calculate it by OpenStaticAnalyze tool.

Table 3: Object Oriented Length Measure

Class	DIT
Aboutus	2
Back_Button	0
Basic_Frame_Duplicity	1
ButtonSound	0
Capitalization	2
Check_extension	0
Duplicity_Folder	2
Duplicity_Random_File	2
File_read	0
Frame_Container_Template	0
Highlight_text	0
Home	0
MainClass	0
Panel_BackButton_Template	1
Save_file	0
Sorting_result	0
Spell_Check	2
SplashScreen	0
User_GuideLines	2

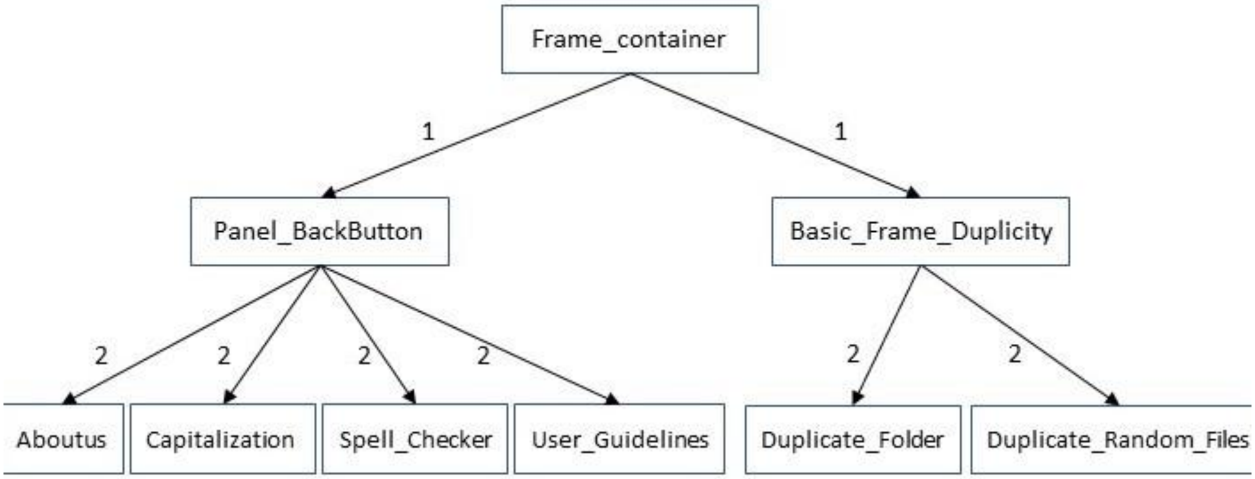

Figure 31: Depth of Inheritance

4. Object-Oriented Reuse Measurement

We can measure Object-Oriented reuse measurement by NOC (Number of Children).

NOC is computed by counting the number of immediate successors (subclasses or subinterfaces) of a class or interface.

Measurement Process: Here we count the number of children of each class by looking at the Inheritance tree.

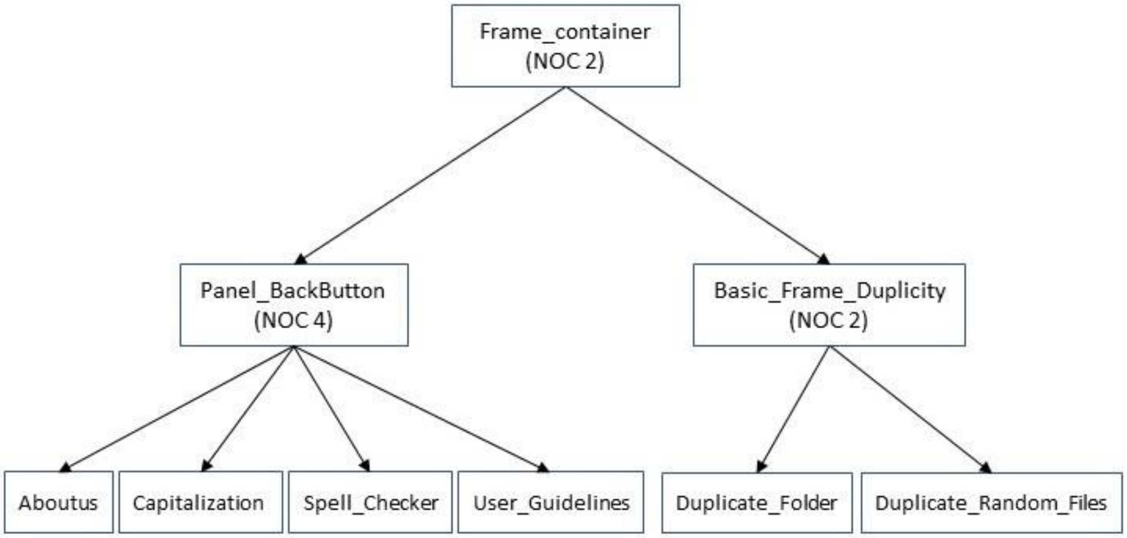

Figure 32: Number of Children

Class	NOC
Aboutus	0
Back_Button	0
Basic_Frame_Duplicity	2
ButtonSound	0
Capitalization	0
Check_extension	0
Duplicity_Folder	0
Duplicity_Random_File	0
File_read	0
Frame_Container_Template	2
Highlight_text	0
Home	0
MainClass	0
Panel__BackButton_Template	4
Save_file	0
Sorting_result	0
Spell_Check	0
SplashScreen	0
User_GuideLines	0